On The Development of Visual Object Memory: The Stay/Go Decision Problem

Clayton T. Morrison
Computer Science
University of Massachusetts
Amherst, MA 01003
clayton@cs.umass.edu

Paul R. Cohen
Computer Science
University of Massachusetts
Amherst, MA 01003
cohen@cs.umass.edu

Paola Sebastiani
Mathematics & Statistics
University of Massachusetts
Amherst, MA 01003
sebas@math.umass.edu

Abstract

A developing memory requires a mechanism for deciding how much information to gather, based on what is currently represented in memory. That is, we need to know when we have seen enough to say we have or have not seen this before, or that we need to continue collecting data. We present a novel statistical approach to this decision mechanism. This serves as the foundation for a simple visual object memory. We present results from simulations showing that the statistical measure can serve as the basis of the stay/go decision process.

1 Introduction

Consider a robot faced with the task of learning to distinguish objects in its environment. As the robot moves around its environment it makes visual contact with objects, some of which it has seen before and some of which are novel. Once it makes visual contact, the robot looks at an object from several angles before deciding to move on in search of other objects. This paper develops a statistical framework for the decision to stay and continue looking at an object or go and look at another object. We call this the stay/go problem. This paper does not try to solve the problem in an optimal way, rather, it provides a simple method that keeps the robot looking at an object until it is reasonably certain that additional views will not help it discriminate the object from others it knows. The set of views that have been observed are the foundation of a visual object memory.

2 Object Representation: Curvature Scale-Space

Our robot is a Pioneer II with a Sony pan-tilt-zoom camera. Each image of an object is processed by an algorithm that generates a curvature scale-space (CSS) representation. CSS diagrams like the one in

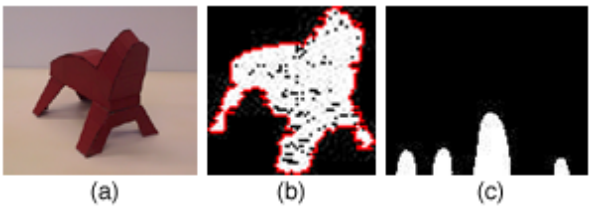


Figure 1: Curvature Scale Space representation: (a) Original image of object, (b) Extracted pixels and border of tracked object, (c) Curvature Scale Space diagram.

Figure 1.c represent how the curvature of each point on a silhouette of the object changes with repeated smoothing [1]. The horizontal axis (often denoted by u) represents the length of the silhouette curve, and the vertical axis (often denoted by σ) represents the degree of smoothing applied to the curvature. The shift from black to white along each horizontal line of the diagram indicates a shift from positive to negative change in the slope of the tangent line to the silhouette curve. Each peak in a CSS diagram represents an "appendage" in the original silhouette (Fig. 1.b). Suppose one has the CSS diagram δ of a new image and wishes to identify the object in the image by comparing δ to other CSS diagrams in a corpus and retrieving their associated images. It has been shown [2, 3] that the information in CSS diagrams is usually sufficient to retrieve images that closely match the outline shape of the new image, despite differences in scale and orientation.

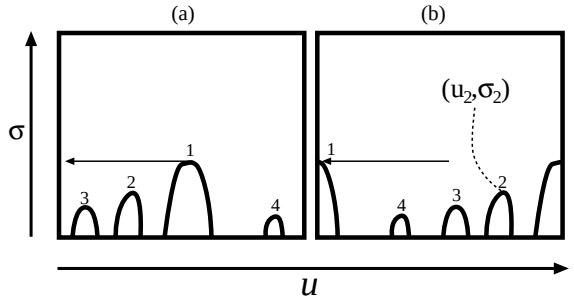
CSS diagrams can be represented as points in a high-dimensional space as follows: Each peak in a diagram has a horizontal and vertical coordinate, so a

diagram with N peaks can be represented as a single point in a space of $2n$ dimensions. However, not all CSS diagrams have the same number of peaks, and the location of peaks depends on the rotation of the object in the image. We fix the latter problem by selecting the tallest peak in the CSS diagram and shifting it to the left-hand border. Since the horizontal axis of the CSS diagram represents the length of the silhouette border, the left and right edges of the horizontal axis are actually adjacent – i.e., the horizontal axis is circular. Any peaks to the left of the tallest peak before shifting "wrap around" to the right side during shifting (Figure 2.a and 2.b). Once a CSS diagram has been shifted, the next task is to tackle the problem of variations in the numbers of CSS peaks. We choose k, the number of peaks we will represent. k may be greater or smaller than the actual number of peaks in any given CSS diagram. The multidimensional point representation of the CSS diagram, called the *canonical representation*, is then $2k$-1−dimensions (−1 because the horizontal coordinate of the largest peak after shifting is always 0, as is the case for coordinate u_1 of peak 1 in Figure 2). Finally, the horizontal position of each peak fills the first $k-1$ dimensions of the canonical representation, and the remaining k dimensions are filled by the vertical values of each point. These are filled in order of the largest to smallest represented CSS diagram peaks. Figure 2 shows an example of this canonicalization; note that the numbering of the peaks follows their order by height, which in turn is reflected in the order in which the values appear in the canonical representation. Had there been six CSS points, the 6^{th} (and therefore smallest) peak would not be represented in the canonicalization.

In summary, images of objects are first represented as CSS diagrams and then as points in $2k-1$-dimensional space (representing the k highest peaks of the diagram).

3 Object Labels and the Stay/Go Decision

As soon as the robot makes visual contact with some object it generates a unique label; it attaches the label to the CSS diagrams of every image it collects while remaining in continuous visual contact. If the robot observed the same object again, at a later time, the images in this set get a new label. In general there are several sets of uniquely-labeled CSS diagrams for each object. When the robot collects 25 images of a dog, then 30 of a man, then another 20 of the original dog, its memory contains two sets of CSS diagrams of the dog, each with a unique label. How big should the second set of dog images be? If additional images



(c) $(u_2, u_3, u_4, 0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, 0)$

Figure 2: Deriving a *canonical representation* of a CSS diagram: (a) Initial CSS diagram, (b) Shifted CSS diagram, with example (u, σ) coordinate for peak 2, (c) canonical representation of $k = 5$ possible CSS points; with only four actual peaks, the remaining dimensions of the canonical representation have 0-values.

cannot help the robot decide either that the object is different from the man or that it is the same as the first dog, then the robot should stop collecting images.

4 A Model of the Stay/Go Decision

In general the stay/go decision concerns the value of additional data: One should stay and collect more data as long as they are valuable, otherwise one should go. The value of data depends on what one intends to *do* with the data; specifically, the value of additional images of an object depends on what the robot will do with the images. Yet we do not want the robot's stay/go strategy to depend on the specifics of a *particular* task. We want the value of images to depend on those the robot already has in memory and on the new images it collects. We want the existing and new images to be put to work in an extremely general task with a measure of merit ϕ, and we want ϕ to reach a maximum after the robot collects a finite number of new images. When ϕ reaches its maximum (or minimum) value, the sample of new images is as big as it should be — making it larger will not increase (or decrease) ϕ, the robot's score on its task. At this point, the robot should stop collecting images of the current object, because these images will not improve its score.

Let us develop a version of ϕ for a simple problem and then show how to extend it to the robot's stay/go decision. Suppose one already has a sample of a random variable x_a with mean $\overline{x}_a$, variance s_a^2, and sample size n_a. Sample a is analogous to a set of identically-labelled CSS diagrams in memory. Now,

one starts to collect new data and accumulate it in sample b. This sample is analogous to what we call new images, above. How many data should one collect, that is, how big should n_b become? Can we formulate a task that, when performed on samples a and b and evaluated with function ϕ, has a maximum value of ϕ at some value n_b? Here is one: We play a game that has two conditions, one in which samples a and b are treated as different, and one in which they are appended into a single sample, and we ask how much better can we play the game in the former condition than the latter. In the first condition, an element x is drawn at random from sample a or sample b, you are told which sample it is from, and invited to guess its value. You are assessed an error, which is the squared difference between your guess and x. To minimize this error your guess should be $\overline{x}_a$ (or $\overline{x}_b$, depending on which sample x comes from). Suppose this game is repeated n_a times with elements of sample a and n_b times with elements of sample b. Then the expected total error assessed against you is:

$$\sum_{i=1}^{n_a}(x_i - \overline{x}_a)^2 + \sum_{j=1}^{n_b}(x_j - \overline{x}_b)^2 \tag{1}$$

We denote these sums SS_a and SS_b. Note that these sums of squares are related to the variances s_a^2 and s_b^2 by $SS_a = n_a s_a^2$ and $SS_b = n_b s_b^2$.

In a second condition of the game, samples a and b are first appended into a single sample g with mean $\overline{x}_g$. The guessing is repeated $n_g = n_a + n_b$ times, with expected error:

$$SS_g = \sum_{i=1}^{n_a}(x_i - \overline{x}_g)^2 + \sum_{j=1}^{n_b}(x_j - \overline{x}_g)^2 \tag{2}$$

Clearly, if a and b are very similar samples in their means and variances, then the value of $SS_a + SS_b$ will be very similar to SS_g; conversely, if a and b are different, then $(SS_a + SS_b) < SS_g$. The difference $SS_g - (SS_a + SS_b)$ can be interpreted as a reduction in errors obtained by making a distinction between samples a and b, as opposed to treating them as one undifferentiated sample g. When we express this reduction in errors as a proportion of SS_g, we obtain the desired function ϕ:

$$\phi = \frac{SS_g - (SS_a + SS_b)}{SS_g} \tag{3}$$

One can see that ϕ has a maximum and that it depends on n_a and n_b (for now, suppose the variances s_a^2 and s_b^2 are equal, we will generalize this in a moment): Clearly, when n_b is much smaller than n_a, SS_g is dominated by SS_a, and, conversely, when n_b is much larger than n_a, SS_g is dominated by SS_b. As SS_g becomes dominated by either SS_a or SS_b the numerator of Equation 3 approaches zero. In fact, when $s_a^2 = s_b^2$, the maximum value of ϕ occurs when $n_a = n_b$. More generally, when the variances of a and b may be unequal, we can find the value of n_b at which ϕ is maximum in the standard way by differentiating Equation 3 and finding the value of n_b at which that function is zero. Equation 3 can be simplified by using the decomposition

$$SS_g = SS_a + SS_b + \frac{(n_a n_b(\overline{x}_a - \overline{x}_b)^2)}{n_g} \tag{4}$$

so that

$$\phi = \frac{(n_a n_b/n_g)(\overline{x}_a - \overline{x}_b)^2}{SS_a + SS_b + (n_a n_b/n_g)(\overline{x}_a - \overline{x}_b)^2} \tag{5}$$

To maximize this function in n_b is difficult because SS_b and $\overline{x}_b$ are functions of n_b, but one can make some approximations. As n_b increases, $(\overline{x}_a - \overline{x}_b)^2$ becomes approximately constant, and provided n_b and n_a are not too small we can approximate both by the sample variances: $SS_a = n_a s_a^2$ and $SS_b = n_b s_b^2$. In this way, the value of n_b that maximizes ϕ is the solution of the equation

$$n_b^2 s_b^2 = n_a^2 s_a^2 \tag{6}$$

When the variances are the same, ϕ is maximized when $n_a = n_b$ and in general when

$$n_b = n_a \frac{s_a^2}{s_b^2} \tag{7}$$

To recap, we drew an analogy between the stay/go decision and the following statistical question: Given a sample a of size n_a with variance s_a^2, how many data should we collect into a sample b for a performance metric ϕ to reach its maximum value? The metric in this case is the reduction in errors that one achieves by treating samples a and b as different (Equation 3). More specifically, if one repeatedly guesses the value of a datum x, then ϕ is the reduction in errors (the difference between guess_i and x_i) due to knowing whether x came from sample a or b. Said differently, ϕ is the value of treating a and b as if they came from different populations. If, in fact, a and b are drawn from different populations, then ϕ will have a nice, clear peak when $n_b = n_a(s_a^2/s_b^2)$, otherwise ϕ will hover around zero for all values of n_b.

Mapping this analogy back to the robot's stay/go decision, the robot should collect a sample of images b

as long as there is value in treating the images in a and b as if they are images of different objects. When they are, in fact, images of different objects, then ϕ will have a well-defined peak at $n_b = n_a(s_a^2/s_b^2)$, otherwise ϕ will hover around zero for all values of n_b.

5 The Stay/Go Decision Methods

It is easy to implement this stay/go strategy when images of objects are represented as CSS diagrams. Recall that the robot's memory contains sets of CSS diagrams and all the diagrams in a set represent images of the same object. Recall, too, that each CSS diagram is transformed to a canonical representation point in $2k$-1−dimensional space. For a given set, a, we can calculate both its centroid $\overline{x}_a$ and SS_a, the sum of squared distances between each element of a and $\overline{x}_a$. This is all the information we need to calculate ϕ as in Equation 3.

There are two general approaches to implementing the stay/go decision process based on ϕ, each of which has two variations (for a total of four methods). The *analytic* approach uses Equation 7, the anticipated number of views needed to achieve a maximal value for ϕ. In Equation 7, sample a corresponds to the set of CSS view representations in memory. The size, n_a, is already known, as well as s_a^2. The difference between the two analytic approaches is in how s_b^2 is calculated, where sample b is the set of new views of the current object the robot is observing. The *Analytic-Complete* method calculates s_b^2 based on the complete set of views for the currently experienced object. Clearly this method can only be implemented in simulation, as it requires the set of possible views for a particular object to be known *a priori*. For the simulations below, this method provides a baseline of performance. The *Analytic-Estimate* method estimates s_b^2 based on the current sample of views. The estimate for the current object being observed will gradually change as new views are acquired.

The other group of stay/go approaches is *empirical* because the decision to go is based on the current shape of the developing ϕ curve as views are collected. As noted, ϕ will either grow to a peak and then decrease as more views are experienced, or ϕ will hover around zero (Figure 3). In the empirical method, a peak-finding test sweeps a window across the ϕ curve, from left to right, looking for the first instance in which the average ϕ values to the left and right of the window are lower than the average of the window center. This will find a peak if one exists. Such a peak is found for ϕ-curve $A1$ in Figure 3 (several peaks are possible for $A1$, but the decision to go would be based on the first that is found). For ϕ-curve $A2$, however, there is not

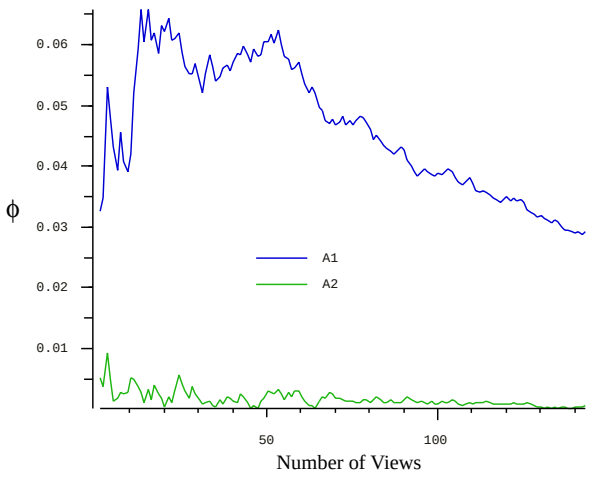


Figure 3: Example of curves generated by plotting values of ϕ as views are accumulated. The two curves represent ϕ values that result from comparing the sets of views for objects A1 and A2 (both in memory) to the current object.

enough of a peak present for the window test. Instead, an additional test must be defined. The choice of test makes the difference between the two variations in the empirical method. In both variations, ϕ is regressed on n_b. The slope of the regression line is expected to be positive when ϕ has a peak and zero otherwise. The *Empirical-Threshold* method calls the regression line flat (i.e., ϕ does not peak) when its slope is less than some threshold near zero. The *Empirical-Confidence* method, on the other hand, calls the regression line flat if a confidence interval for the slope around the line contains zero.

In sum, we have four stay/go decision methods: Analytic-Complete, Analytic-Estimate, Empirical-Threshold, Empirical-Confidence. We now discuss their performance in a simulator using actual object images.

6 The Stay/Go Simulator

Recall that a CSS diagram represents curvature features of an image. We constructed five 3-dimensional objects with a variety of different surface features, allowing for both similarity and variance among CSS diagrams of images taken from different perspectives of the objects. The five object-shapes resemble a thin dog, a four-legged arch, a dog with two heads, a man, and a four-legged object with a round body (a "rotund" dog). We then collected 7 sets of images taken from a variety of perspectives of the objects, all from

the level of the Pioneer II camera angle. Three of the image sets were taken from the thin dog; one of these consisted of a complete rotation around the dog, the other two only half rotations around either side. Each set contained a mean of 34 images.

A simulator was constructed to test the stay/go decision methods. At the beginning of each *epoch*, one of the 7 image sets is selected at random, and the simulator is presented with views randomly sampled from that set. As described in Section 3, all of the views from the selected set are assigned a unique label chosen at the beginning of the epoch. At each view presentation, the simulator decides whether to stay and see another view from that set, or go, terminating that epoch. The stay/go decision is calculated by comparing the current set of views against each set of previously experienced object views stored in memory. The decision to go cannot be made until the ϕ curve for each object in memory passes the decision method's test. After the completion of each epoch, the views sampled during the epoch are added to memory. For the experiments we conducted, each of which consisted of 100 epochs, the simulator's memory was seeded with the complete set of unique views from one of the 7 image sets.

7 Results

We tested the simulator with each of the four stay/go methods. Figures 4-6 plot the number of views before going for each epoch. For clarity, we use the Analytic-Estimate curve as a baseline to compare with each of the other methods' curves. Figure 4 shows that the Analytic-Complete and Analytic-Estimate methods are very similar, with the Analytic-Complete increasing at a slightly faster rate. To understand why both of these curves increase linearly with each epoch, observe that the size of the set of elements for the object in memory, n_a, appears in the calculation of Equation 7. That is, for each comparison, the number of views in a set from memory directly influences the go decision. However, the decision to go cannot be made until the comparison with *each* object in memory satisfies the go criterion. This means that the required number of views will be influenced by the largest set of views in memory. This is the key shortcoming of the analytic methods: as long as the number of objects in memory keep accruing, the mean number of views will increase.

To avoid this problem, we turn to the empirical methods, which depend not on the size of the samples but on the behavior of ϕ. The naive Empirical-Threshold method exemplifies a first-pass solution to this problem. Here, the threshold was set to slightly

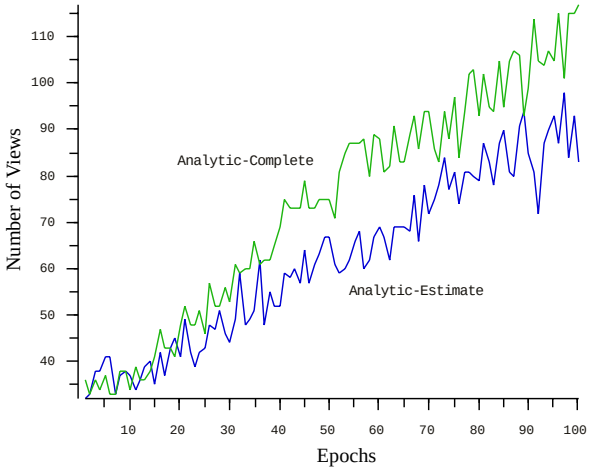


Figure 4: Comparison of Analytic-Estimate and Analytic-Complete.

above zero (0.015), in order to catch those ϕ curves with slopes close to zero, but to avoid catching ϕ curves that may eventually have peaks. Figure 5 shows that this threshold is not desirable. At least for runs of under 100 epochs, the analytic methods perform much better. Empirical-Threshold does, however, have one positive property: despite the large variance in number of views required from one epoch to the next, the overall number of views required does not increase at the rate the analytic methods do. To see this more clearly, we have treated each method's results as a scatterplot and calculated linear regression lines for each. These are summarized in Table 1, which include the overall mean number of views per epoch, the standard deviation (s.d.) of each mean, and the linear regression line slope, along with that slope's confidence interval. The Empirical-Threshold slope is 0.34, which is significantly less than the two analytic methods.

The last method, Empirical-Confidence, provides a much better approach to determining the slope of ϕ curves that are close to flat. In this method, the confidence interval of the linear-regression line slope of the ϕ curve is also calculated, and if a slope of 0 falls within the interval, then the ϕ curve is considered essentially flat. Figure 6 plots the Empirical-Confidence results. Not only does it maintain a slope that is close to horizontal, the number of views required for each epoch is significantly less than any of the other methods. Of the four methods, Empirical-Confidence does the best job of differentiating empirically curves that have peaks from those that are flat.

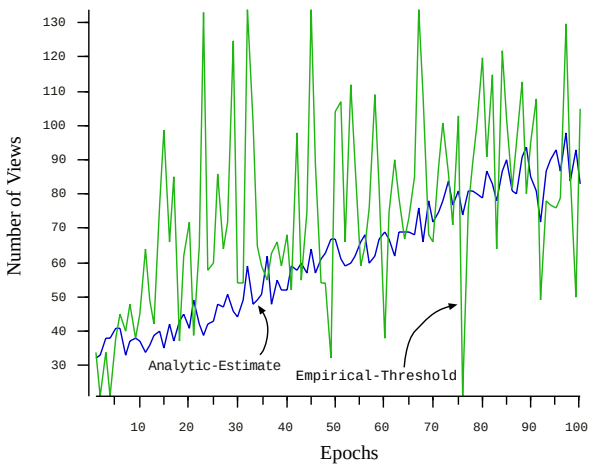


Figure 5: Comparison of Analytic-Estimate and Empirical-Threshold.

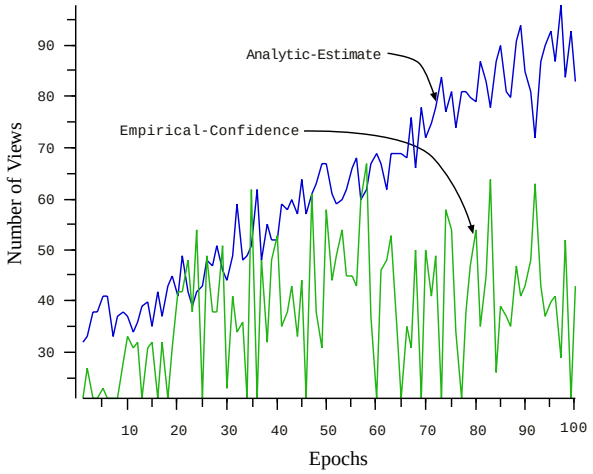


Figure 6: Comparison of Analytic-Estimate and Empirical-Confidence.

method	*mean*	*s.d.*	*regression*	
			slope	*c.i.*
Analytic-Comp	73.90	24.64	0.83	0.0040
Analytic-Est	61.93	18.07	0.60	0.0035
Empirical-Thresh	69.17	22.97	0.34	0.0167
Empirical-Conf	38.32	12.34	0.15	0.0093

Table 1: Summary Statistics of Results

8 Summary and Conclusions

The Empirical-Confidence decision method has two interesting properties: (1) even early on it tends to require fewer views than the analytic methods do, and, more importantly, (2) the number of views required within each epoch remains relatively stable as the number of objects represented in memory increases. This makes the Empirical-Confidence method, based on the behavior of the ϕ statistic as views are accrued, an attractive approach to a general stay/go solution.

Up to this point, the sets of views experienced for each epoch have simply been stored in memory as individual sets. But many of these sets are actually sampled from the same object. The next step is to investigate methods for merging sets of views that are similar, and to evaluate how such merging interacts with the stay/go criterion over time. The goal is a simple visual object memory that maintains classes in memory that accurately represent classes of objects in the environment. Such a memory will be a significant step towards an autonomously developing agent.

Acknowledgments

This research was supported by DARPA under contract Number DASG60-99-C-0074. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA or the U.S. Government.

References

[1] F. Mokhtarian and A. K. Mackworth, "A theory of multiscale, curvature-based shape representation for planar curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-14, No. 8, pp. 789-805, August 1992.

[2] F. Mokhtarian, S. Abbasi and J. Kittler, "Efficient and Robust Retrieval by Shape Content through Curvature Scale Space," *Proc. International Workshop on Image Databases and Multimedia Search*, pp. 35-42, 1996.

[3] S. Abbasi and F. Mokhtarian, "Affine-Similar Shape Retrieval: Application to Multiview 3-D Object Recognition," *IEEE Transactions on Image Processing*, Vol. 10, No. 1, pp. 131-139, January 2001.